Assignment 1 – A: Group them up - 2016CS150

Identification of the Data Structure: The given scenario asks to divide some given data sets into two sides in a recursive way. One is front (right) and the other is back (left). To so this we can use trees as well as list or arrays also. But insertion, deletion, searching of an unsorted array will takes O(n) and to do asked scenarios may take $O(n^2)$ or for arrays or lists. So the best way to do given insertion and searching by using a Binary search tree. It will take O(log n) for all insertions and searching. As well as we can divide two sets easily to left sub tree or right sub tree. So the most relevant and less complex data structure is Binary search tree. I simply modified it for further calculations and insertions.

Solving the problem: I used a list to store given inputs such as number of lines, which line should be considered and coordinates of the each lines. By importing "re" from python library and by using "re.findall", can identify all integers of a given string and by using them I converted all string integers to integers. When inserting new line into the tree we have to consider some special cases. There is a function to calculate slope (m) and Y intercept (c) for every given lines and by using it, I calculate m and c for every lines. And there is another function to find intersection of given two lines. First I'll briefly explain what the basic functions in my program are and what they basically do. After that we'll consider about intersections and special cases of intersections.

Basic functions of the program and what they really do: There is a class named as node, which is the basic element of the tree. And tree class has some methods to insert nodes, some basic traversal methods for further outputs. No delete functions as it's not necessary for this program. And there is a class called as calculations, and there is a method to find slope and Y intersect. And there is another class called as Positions, which will help me to find the positions of the given straight lines. The method mEqual and cEqual will return a Boolean value if the given conditions are true or false. And the leftOright method will tell me if the given lines are not intersected then if that line is in left side or the right side of the other line. To do that I used slopes and Y intersect point of the given lines and there is another function to find intersections, which will describe more about that function in next paragraph. And the main function to run the whole program, store some data and print the outputs.

Special cases when finding intersection: There are four ways to intersect lines. One way is the multiplication of both slopes ($m1 \times m2 < 0$) are negative and the others are multiplication of both slopes are positive, zero or infinity. When $m1 \times m2 < 0$, then $m1 < 0$ or $m2 < 0$. Let's consider c1 as given line's intercept and m2 as variable line's intercept. If $c1 > c2$ and m1's maximum Y coordinate is between m2's Y coordinates then there is an intersection. As well as if $c2 < c1$ and m1's minimum Y coordinate is between m2's Y coordinates then there is an intersection. I used max and min functions in math library to calculate them. If $m1 \times m2 > 0$ then both m's are positive or negative. Then the both lines intersect if the intersection point is in range of given variable lines X coordinates. Then the next situation is if one line's slope is infinity. When this occurs if there is an intersection between those lines, the intersection points X coordinate should be in range of the other lines X coordinates if necessary Y coordinates also. The last way to happens an intersection is one lines slope is zero. When this occurs we have to find the intersecting point and whether is it in the range of the variable lines X coordinates if necessary Y coordinates also. If it is, then there is an intersection. So those are the special cases for intersections.

Special cases when inserting elements into the tree: So now we can identify whether given two lines intersect or not by using above mentioned method. When inserting elements into the tree we have to

consider two parts mainly. One is if those lines are intersected or the other thing is not intersected. Let's consider about if them are not intersected. Then we have to think about several conditions. There can be lines with same slope and they will never intersect. And we have to think about the slope also, whether it's infinity, zero, and negative or positive. So lines with same slop (except infinity slopes) will be put into right or left side of the tree by checking their slopes and Y intersect points. If intersect point of the given line is higher than the main line, it will be added to left and else it will be added to right sub tree. If both intersection points are equal then it won't added as both lines represent a single line. If main lines slope is infinity and both lines slopes are equal then according to X coordinates I put them into tree. If the value is less than the X coordinate value of main slope it will be added to left or else it will be added to right. Those are the cases, the lines have equal slopes. Then let's consider about the slopes that has different slope and not intersected. In here let's consider both lines slopes are not equal to infinity. If m1 x m2 of those lines are greater than zero, it means both lines are in kind of same angle. Then by considering slope and Y intersect we can put the line to right or left. If the main lines slope is equal to zero then we have to consider the Y coordinate of the main line and whether it's less than to other lines Y coordinate. If yes it should be added to left and else to right. If the other lines slope is equal to zero if main lines maximum X coordinate's value is less than the variable lines minimum X coordinates then it should added to right side of the tree or else should be added to left side of the tree. Then if one of those slope is less than zero, it means m1 x m2 < 0, if the X coordinates of the given line are higher than main lines X coordinates it should be added to right, unless left. Then let's consider about when slope becomes infinity. First if main lines slope becomes infinity, the maximum value of X coordinates' is less than main lines any X coordinates' then it should be added to left side of the tree, unless should be added to right sub tree. So those are the main scenarios that can be occurred when selecting the side of the given line. And of course if there is any intersection between those given lines, that line should be added to both side of the tree. So by here we know how to find any intersection and how to insert data into the tree. And we should find all left and right lines for each sub roots of the tree. For an example imagine p1, p2, p3 are three different lines and p2 and p3 is in right side of p1 and p3 in left side of p2. Then imagine p1 is main line. Then p2 and p3 will definitely go to right side of the tree. The right most child of root is p2 and we should check p3 with p1 and p2. Then p3 goes to right of the tree but left for the right sub tree. Likewise we should do this method recursively until there is no line to input. After all are inserted the tree is ready to do other calculations, searches etc.

Importations and language: I used python 3.3 to implement above scenario. As well as I used some python libraries to make some functions more efficient. Some of them are re.findall from re library and max, min from math library.

Special cases, what I couldn't do and how to get output: If all the lines are in a same line but different point, then the tree is empty and couldn't do anything. And I couldn't find the lines that are face each other. And the last node of the right sub tree will be the line that face front for all lines. And by calling pre order or post order traversal (depends on which side you choose to insert front and back lines to the tree) to print whole lines of the tree that are in back side or front side for the given line.

Complexity: The tree has O (log n) complexity for insertion, traversal and deletion etc. And there is a 'for' loop which will run 'n' times for given 'n' and in 'for loop' I call tree insertion. And whole complexity should be O (n log n) for my program (not considering the complexity of 'if' clauses).